What is claimed is:

1. (Currently Amended) A processor comprising:

a processing core to execute a trace having one or more lines of one or more micro-operations; and

an optimizer to optimize the trace based on runtime information collected by the processing core during a previous execution of the trace ~~upon each execution of the trace by the processing core.~~

2. (Original) The processor of claim 1, wherein the optimizer is a pipelined optimizer.

3. (Original) The processor of claim 1, further comprising a trace cache to store a trace from said optimizer.

4. (Currently Amended) The processor of claim 3, further comprising:

an instruction cache to store static code received from a compiler via a memory;

a ~~mite~~ macro-instruction translation engine to translate the static code into micro-operations; and

a fill buffer to build a trace from the micro-operations.

5. (Original) The processor of claim 4, further comprising a trace queue to store one or more lines of one or more traces from the fill buffer and one or more lines

from one or more traces from the trace cache.

6. (Original) The processor of claim 5, further comprising an allocator to send traces from the trace queue to the processing core and the optimizer.

7. (Original) The processor of claim 1, wherein the processing core is an out of order processing core.

8. (Original) The processor of claim 1, wherein the optimizer is to track optimizations executed on a specific trace.

9. (Original) The processor of claim 1, wherein the optimizer is to pack the trace after optimization.

10. (Original) The processor of claim 9, wherein the optimizer is to pack the trace by optimizing two consecutive lines of a trace simultaneously.

11. (Original) The processor of claim 10, wherein the optimizer is to use an alternating offset to determine the two consecutive lines of the trace to optimize together.

12. (Original) The processor of claim 1, wherein optimizations includes at least one of a group of optimizations consisting of call return elimination, dead code elimination, dynamic μop fusion, binding, load balancing, move elimination, common

sub-expression elimination, constant propagation, redundant load elimination, store forwarding, memory renaming, trace specialization, value specialization, reassociation, and branch promotion.

13. (Cancelled)

14. (Currently Amended) The processor of claim ~~13~~3, wherein the runtime information is appended to the trace in the trace cache.

15. (Currently Amended) The processor of claim ~~13~~3, further comprising a runtime information buffer to store the runtime information, the runtime information buffer mapped to the trace cache to match the runtime information with the trace.

16. (Currently Amended) An optimization unit comprising:

an input to receive a trace ~~each time the trace is~~ also being sent to a processing core to be executed; and

an optimizer to optimize the trace based on runtime information collected by the processing core during a previous execution of the trace.

17. (Original) The optimizing unit of claim 16, wherein the optimizer is a pipelined optimizer.

18. (Original) The optimizing unit of claim 16, further comprising an output

connected to a trace cache to store an optimized trace after optimization by the optimizer.

19. (Original) The optimizing unit of claim 16, wherein the input is connected to an allocator, the allocator to send traces from a trace queue storing optimized and unoptimized traces to the processing core and the optimizer.

20. (Original) The optimizing unit of claim 16, wherein the optimizer tracks optimizations executed on a specific trace.

21. (Original) The optimizing unit of claim 16, wherein the optimizer packs the trace after optimization.

22. (Original) The optimizing unit of claim 21, wherein the optimizer packs the trace by optimizing two or more consecutive lines of a trace simultaneously.

23. (Original) The optimizing unit of claim 22, wherein the optimizer uses an alternating offset to determine the two or more consecutive lines of the trace to optimize.

24. (Original) The optimizing unit of claim 16, wherein optimizations includes at least one of a group of optimizations consisting of call return elimination, dead code elimination, dynamic μop fusion, binding, load balancing, move elimination, common sub-expression elimination, constant propagation, redundant load elimination, store

forwarding, memory renaming, trace specialization, value specialization, reassociation, and branch promotion.

25. (Cancelled)

26. (Currently Amended) A method comprising:

executing a trace in a processing core; and

simultaneously optimizing the trace based on runtime information collected by the processing core during a previous execution of the trace. ~~each time the trace is executed~~.

27. (Original) The method of claim 26, further including storing the trace after optimization in a trace cache.

28. (Original) The method of claim 27, further including storing unoptimized traces to be processed and optimized.

29. (Original) The method of claim 28, further comprising:

storing static code from a compiler;

translating the static code into micro-operations; and

building an unoptimized trace from the micro-operations.

30. (Original) The method of claim 26, wherein the processing core is an out

of order processing core.

31. (Original) The method of claim 26, further including tracking optimizations executed on a specific trace.

32. (Original) The method of claim 26, further including packing the trace after optimization.

33. (Original) The method of claim 32, wherein the trace is packed by optimizing two or more consecutive lines of a trace simultaneously.

34. (Original) The method of claim 33, further including using an alternating offset to determine the two or more consecutive lines of the trace to optimize.

35. (Original) The method of claim 26, wherein optimizing includes at least one of a group of optimizations consisting of call return elimination, dead code elimination, dynamic μop fusion, binding, load balancing, move elimination, common sub-expression elimination, constant propagation, redundant load elimination, store forwarding, memory renaming, trace specialization, value specialization, reassociation, and branch promotion.

36. (Cancelled)

37. (Currently Amended) The method of claim ~~36~~ 26, further including appending the runtime information to the trace.

38. (Currently Amended) A system comprising:

a memory to store a trace;

a processor coupled to said memory to execute a trace in a processing core and to simultaneously optimize the trace based on runtime information collected during a previous execution of the trace. ~~each time the trace is executed.~~

39. (Original) The system of claim 38, wherein the processor has an out of order processing core.

40. (Original) The system of claim 38, wherein the processor tracks optimizations executed on a specific trace.

41. (Original) The system of claim 38, wherein the processor packs the trace after optimization.

42. (Original) The system of claim 41, wherein the trace is packed by optimizing two or more consecutive lines of a trace simultaneously.

43. (Original) The system of claim 42, wherein an alternating offset is used to determine the two or more consecutive lines of the trace to optimize.

44. (Original) The system of claim 38, wherein optimizing includes at least one of a group of optimizations consisting of call return elimination, dead code elimination, dynamic μop fusion, binding, load balancing, move elimination, common sub-expression elimination, constant propagation, redundant load elimination, store forwarding, memory renaming, trace specialization, value specialization, reassociation, and branch promotion.

45. (Cancelled)

46. (Currently Amended) A set of instructions residing in a storage medium, said set of instructions capable of being executed by a processor to implement a method for processing data, the method comprising:

executing a trace in a processing core; and

simultaneously optimizing the trace based on runtime information collected during a previous execution of the trace. ~~each time the trace is executed~~.

47. (Original) The set of instructions of claim 46, further including tracking optimizations executed on a specific trace.

48. (Original) The set of instructions of claim 46, further including packing

the trace after optimization.

49. (Original) The set of instructions of claim 48, wherein the trace is packed by optimizing two or more consecutive lines of a trace simultaneously.

50. (Original) The set of instructions of claim 49, further including using an alternating offset to determine the two or more consecutive lines of the trace to optimize.

51. (Original) The set of instructions of claim 46, wherein optimizing includes at least one of a group of optimizations consisting of call return elimination, dead code elimination, dynamic μop fusion, binding, load balancing, move elimination, common sub-expression elimination, constant propagation, redundant load elimination, store forwarding, memory renaming, trace specialization, value specialization, reassociation, and branch promotion.

52. (Cancelled)

53. (Currently Amended) The set of instructions of claim ~~52~~ 46, further including appending the runtime information to the trace.